3. What is an SSRF Attack and How Does it Work?

What is an SSRF Attack?

SSRF (Server-Side Request Forgery) is a web application vulnerability where an attacker forces a server to make unauthorized requests on their behalf.

The goal is to exploit the server's access to internal or external systems, bypassing firewalls or authentication barriers to access sensitive data, perform malicious actions, or even escalate privileges.

Key Characteristics of an SSRF Attack

1. **Server as a Proxy:** The attacker manipulates the server into making requests they wouldn't otherwise be able to make directly.
2. **Targets Internal and External Systems:**
   - **Internal Systems:** Exploit private IPs (e.g., `127.0.0.1`) or services behind a firewall.
   - **External Systems:** Interact with APIs or steal cloud metadata.

How Does an SSRF Attack Work?

1. The Vulnerable Feature

- The web application has a functionality where users can provide a **URL** or some other input that triggers a server-side request.
- Example: Fetching user-provided URLs for:
  - Retrieving images.
  - Sending webhooks.
  - API communication.

---

2. The Attacker Identifies the Entry Point

- The attacker identifies that they can control the URL or resource requested by the server.
  For example:

```
GET /fetch?url=http://example.com/image.png
```

---

3. Crafting a Malicious Request

- Instead of supplying a normal URL, the attacker injects a **malicious URL** to manipulate the server. Examples:
  - **Internal access:**

    ```
    http://127.0.0.1/admin
    ```

  - **Cloud metadata access:**

    ```
    http://169.254.169.254/latest/meta-data
    ```

  - **Port scanning:**

    ```
    http://192.168.1.1:22
    ```

---

4. Server Executes the Request

- The server blindly executes the request **as if it were legitimate**.
- This can result in:
  - Leaking sensitive data (like admin panels or cloud credentials).
  - Interacting with services the attacker isn't authorized to access.
  - Performing malicious actions (like deleting resources or triggering APIs).

---

5. The Attacker Gets the Response

- The server's response reveals sensitive information or confirms that the request succeeded.
- Example outcomes:
  - Accessing an internal admin interface.
  - Stealing cloud credentials:

    ```
    IAM Role: S3-access-full
    ```

  - Mapping internal network services.

A Practical Example

Legitimate Use:

- **URL Input by User:**

  ```
  GET /fetch?url=http://example.com/image.jpg
  ```

- **Server Action:**
  Fetches the image and displays it.

Exploit:

- **Malicious URL Provided by Attacker:**

```
GET /fetch?url=http://127.0.0.1/admin
```

- **Server Action:**
  Fetches the admin page, potentially leaking sensitive information to the attacker.

Why SSRF is Dangerous

1. **Internal Network Access:**
   Attackers can bypass firewalls to access internal systems or APIs.
2. **Cloud Exploits:**
   Attackers can fetch metadata in cloud services (AWS, GCP, etc.), exposing credentials or tokens.
3. **Port Scanning and Enumeration:**
   Use the server to probe other systems for open ports and services.